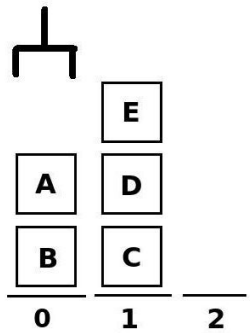
Raúl García Furió A01205785
Antonio Grimaldo A01151618
Ana María González Montoya A01204110

Implementing (Un)informed Search Algorithms



1. Which heuristics did you use for the A* algorithm?

- a_star_cons: For this heuristic we obtain, for each box, the number of piles it needs to move to reach the pile of the goal state. ie. for the above state, assuming the goal state is [[B], [C, D], [A, E]] the heuristic will be : A->2, B->0, C->0, D->0, E->1, **h(s) = 3**

- a_star_incons: For this heuristic we choose the Manhattan distance. For example, in the image above if we want to move A to the container 2 the A -> 3 and the **h(s) = 3** because you do not need to move any other box.

2. Which of the four algorithms searches the least nodes and which one take the most? and why does this happen?

   DFS reach a longer path than BFS because it search through all the child's of a node and if there is not the node that we are searching it generated all that path already.

   A* give us the best path possible but it take a few milliseconds more because it need to check the heuristic.

3. Which algorithms are optimal? Why?

   DFS is optimal just if the tree is limited, in other words, if the tree has more that one goal and the goal is in the left part of the tree the algorithm become non-optimal

   BFS just if the tree is unweighted. it is optimal only if the problem present an uniformed path cost, It requires a lot of memory to be executed

   A* is optimal it will always return the best answer(path), but it time complexity depends directly on the heuristic, if it counts with a heuristic the algorithm will be executed in a linear time, in the worst of the escenarios the complexity will be

exponencial. The biggest problem of this algorithm is the memory space, because it has to save all possible paths of all the nodes

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^{d+1})$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^{d+1})$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.17 Evaluation of search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

4. In your opinion what are the benefits of simpler algorithms versus more complex ones?

Simpler algorithms could give you the correct answer but they can return a solution that is not optimal or that is more complex. But these are easier to implement and understand.

Otherwise more complex algorithm can use more specific data structures and patterns that will be consistent and will give the best answer. But these algorithms are more difficult to implement and to understand.

Reference:

Artificial Intelligence: A Modern Approach (2nd Edition) by Stuart Russell and Peter Norvig, Chapter 3: Solving problems by searching.